Business problem understanding

- Netflix Movies and TV Shows

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
# Load data

df = pd.read_csv("Netflix.csv")
df
```

Out[2]:

	Title	Type	Genre	Release Year	Rating	Duration	Country
0	Title 1	TV Show	Comedy	1955	PG	3 Seasons	Japan
1	Title 2	TV Show	Horror	2020	G	3 Seasons	India
2	Title 3	TV Show	Action	1966	TV-PG	140 min	United States
3	Title 4	Movie	Thriller	2011	PG-13	3 Seasons	Canada
4	Title 5	TV Show	Romance	1959	TV-14	172 min	India
...	...	...	...	...	...	...	...
2995	Title 2996	Movie	Thriller	2007	TV-PG	75 min	Germany
2996	Title 2997	Movie	Drama	2019	G	2 Seasons	Germany
2997	Title 2998	TV Show	Action	1993	R	3 Seasons	Canada
2998	Title 2999	Movie	Drama	1966	PG-13	1 Seasons	Germany
2999	Title 3000	TV Show	Thriller	2015	PG	2 Seasons	United States

3000 rows × 7 columns

Data Exploration

- It helps data scientists understand the dataset, identify patterns, and gain insights before further analysis

In [6]:
```
# It represent the number of rows and columns in the DataFrame

df.shape
```

```
Out[6]: (3000, 7)
```

```
In [8]: # To extract the column names of a DataFrame

        df.columns.tolist()
```

```
Out[8]: ['Title', 'Type', 'Genre', 'Release Year', 'Rating', 'Duration', 'Country']
```

```
In [10]: # Prints information about the DataFrame

         df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3000 entries, 0 to 2999
Data columns (total 7 columns):
 #   Column        Non-Null Count  Dtype 
---  ------        --------------  ----- 
 0   Title         3000 non-null   object
 1   Type          3000 non-null   object
 2   Genre         3000 non-null   object
 3   Release Year  3000 non-null   int64 
 4   Rating        3000 non-null   object
 5   Duration      3000 non-null   object
 6   Country       3000 non-null   object
dtypes: int64(1), object(6)
memory usage: 164.2+ KB
```

Unique-> Returns unique values from a data series

```
In [13]: # Categorical

         df['Title'].unique()
```

```
Out[13]: array(['Title 1', 'Title 2', 'Title 3', ..., 'Title 2998', 'Title 2999',
                'Title 3000'], dtype=object)
```

```
In [15]: # Categorical

         df['Type'].unique()
```

```
Out[15]: array(['TV Show', 'Movie'], dtype=object)
```

```
In [17]: # Categorical

         df['Genre'].unique()
```

```
Out[17]: array(['Comedy', 'Horror', 'Action', 'Thriller', 'Romance', 'Drama',
                'Documentary', 'Sci-Fi'], dtype=object)
```

```
In [19]: # continuous

         df['Release Year'].unique()
```

```
Out[19]:  array([1955, 2020, 1966, 2011, 1959, 2007, 1977, 1971, 2000, 1975, 2021,
                 1986, 1997, 1994, 1996, 1969, 1983, 2023, 1993, 1968, 1965, 1991,
                 2004, 1952, 1992, 1989, 2019, 1999, 1964, 2003, 1981, 2012, 1961,
                 1967, 1973, 1980, 2018, 2016, 2014, 2005, 1970, 1960, 2001, 2015,
                 1954, 1962, 1995, 2006, 1974, 1963, 1950, 2002, 1988, 1951, 1978,
                 1972, 1985, 2010, 2008, 1982, 1953, 1998, 1979, 1984, 1976, 1956,
                 1990, 2022, 1958, 2017, 1987, 1957, 2009, 2013], dtype=int64)
```

```
In [21]:  # Categorical

          df['Rating'].unique()
```

```
Out[21]:  array(['PG', 'G', 'TV-PG', 'PG-13', 'TV-14', 'TV-MA', 'R'], dtype=object)
```

```
In [23]:  # This is Wrong Data type i want to separate Min and Seasons from data

          df['Duration'].unique()
```

```
Out[23]:  array(['3 Seasons', '140 min', '172 min', '68 min', '104 min',
                 '2 Seasons', '1 Seasons', '139 min', '85 min', '129 min',
                 '107 min', '147 min', '120 min', '115 min', '164 min', '175 min',
                 '78 min', '88 min', '133 min', '156 min', '141 min', '76 min',
                 '130 min', '67 min', '128 min', '97 min', '169 min', '137 min',
                 '166 min', '66 min', '89 min', '174 min', '72 min', '138 min',
                 '143 min', '86 min', '168 min', '108 min', '109 min', '124 min',
                 '84 min', '106 min', '153 min', '157 min', '180 min', '90 min',
                 '159 min', '74 min', '65 min', '79 min', '73 min', '103 min',
                 '165 min', '179 min', '113 min', '91 min', '119 min', '155 min',
                 '135 min', '70 min', '177 min', '126 min', '158 min', '122 min',
                 '127 min', '173 min', '125 min', '131 min', '161 min', '114 min',
                 '178 min', '163 min', '132 min', '110 min', '111 min', '81 min',
                 '151 min', '145 min', '92 min', '71 min', '77 min', '100 min',
                 '134 min', '176 min', '171 min', '150 min', '94 min', '61 min',
                 '116 min', '96 min', '148 min', '121 min', '101 min', '102 min',
                 '87 min', '149 min', '146 min', '95 min', '63 min', '160 min',
                 '60 min', '98 min', '75 min', '152 min', '136 min', '123 min',
                 '118 min', '170 min', '83 min', '112 min', '82 min', '80 min',
                 '93 min', '99 min', '167 min', '117 min', '69 min', '144 min',
                 '62 min', '105 min', '154 min', '64 min', '162 min', '142 min'],
                dtype=object)
```

TReat Wrong data Type

```
In [26]:  # Movies (those that contain 'min')
          movies = df[df['Duration'].str.contains('min', na=False)]

          # TV Shows (those that contain 'Season')
          tv_shows = df[df['Duration'].str.contains('Season', na=False)]
```

Replace Min to " "

```
In [29]:  movies['Duration'] = movies['Duration'].str.replace(' min', '').astype(int)
```

```
C:\Users\WELCOME\AppData\Local\Temp\ipykernel_5112\2268893789.py:1: SettingWithCopyW
arning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/u
ser_guide/indexing.html#returning-a-view-versus-a-copy
  movies['Duration'] = movies['Duration'].str.replace(' min', '').astype(int)
```

Replace Season to " " And s to " "

In [32]:
```
tv_shows['Duration'] = tv_shows['Duration'].str.replace('Season', '').str.replace('
```

```
C:\Users\WELCOME\AppData\Local\Temp\ipykernel_5112\2326678699.py:1: SettingWithCopyW
arning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/u
ser_guide/indexing.html#returning-a-view-versus-a-copy
  tv_shows['Duration'] = tv_shows['Duration'].str.replace('Season', '').str.replace
('s', '').astype(int)
```

Combine movies and tv_shows

In [35]:
```
df = pd.concat([movies, tv_shows])
```

In [37]:
```
# continuous

df["Duration"].unique()
```

Out[37]:
```
array([140, 172,  68, 104, 139,  85, 129, 107, 147, 120, 115, 164, 175,
        78,  88, 133, 156, 141,  76, 130,  67, 128,  97, 169, 137, 166,
        66,  89, 174,  72, 138, 143,  86, 168, 108, 109, 124,  84, 106,
       153, 157, 180,  90, 159,  74,  65,  79,  73, 103, 165, 179, 113,
        91, 119, 155, 135,  70, 177, 126, 158, 122, 127, 173, 125, 131,
       161, 114, 178, 163, 132, 110, 111,  81, 151, 145,  92,  71,  77,
       100, 134, 176, 171, 150,  94,  61, 116,  96, 148, 121, 101, 102,
        87, 149, 146,  95,  63, 160,  60,  98,  75, 152, 136, 123, 118,
       170,  83, 112,  82,  80,  93,  99, 167, 117,  69, 144,  62, 105,
       154,  64, 162, 142,   3,   2,   1])
```

In [39]:
```
# Categorical

df['Country'].unique()
```

Out[39]:
```
array(['United States', 'India', 'Japan', 'Australia', 'Germany',
       'South Korea', 'Canada', 'United Kingdom'], dtype=object)
```

In [41]:
```
# Prints information about the DataFrame

df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 3000 entries, 2 to 2999
Data columns (total 7 columns):
 #   Column        Non-Null Count  Dtype 
---  ------        --------------  ----- 
 0   Title         3000 non-null   object
 1   Type          3000 non-null   object
 2   Genre         3000 non-null   object
 3   Release Year  3000 non-null   int64 
 4   Rating        3000 non-null   object
 5   Duration      3000 non-null   int32 
 6   Country       3000 non-null   object
dtypes: int32(1), int64(1), object(5)
memory usage: 175.8+ KB
```

To check which is continous count And categorical

In [44]:
```
continuous = ["Release Year","Duration"]

categorical = ["Title","Type","Genre","Rating","Country"]
```

Generate descriptive statistics of a DataFrame

In [47]:
```
df[continuous].describe()
```

Out[47]:

	Release Year	Duration
count	3000.000000	3000.000000
mean	1985.935333	61.934000
std	21.220617	64.212164
min	1950.000000	1.000000
25%	1968.000000	2.000000
50%	1986.000000	61.000000
75%	2004.000000	120.000000
max	2023.000000	180.000000

In [49]:
```
df[categorical].describe()
```

Out[49]:

	Title	Type	Genre	Rating	Country
count	3000	3000	3000	3000	3000
unique	3000	2	8	7	8
top	Title 3	TV Show	Horror	TV-14	Canada
freq	1	1527	398	441	421

In [51]:
```
# To Check Missing value

df.isnull().sum()
```

Out[51]:
```
Title           0
Type            0
Genre           0
Release Year    0
Rating          0
Duration        0
Country         0
dtype: int64
```

In [53]:
```
# To check duplicate

df.duplicated().sum()
```

Out[53]:
```
0
```

skewness is only meaningful for numerical (continous or count) variables

In [56]:
```
df[continuous].skew()
```

Out[56]:
```
Release Year    0.030949
Duration        0.400881
dtype: float64
```

In [58]:
```
# This is clean Data
df
```

Out[58]:

	Title	Type	Genre	Release Year	Rating	Duration	Country
2	Title 3	TV Show	Action	1966	TV-PG	140	United States
4	Title 5	TV Show	Romance	1959	TV-14	172	India
6	Title 7	Movie	Romance	1977	TV-14	68	United States
7	Title 8	Movie	Comedy	1971	TV-PG	104	Japan
10	Title 11	TV Show	Romance	2021	TV-14	139	Australia
...	...	...	...	...	...	...	...
2993	Title 2994	Movie	Comedy	1969	TV-PG	1	South Korea
2996	Title 2997	Movie	Drama	2019	G	2	Germany
2997	Title 2998	TV Show	Action	1993	R	3	Canada
2998	Title 2999	Movie	Drama	1966	PG-13	1	Germany
2999	Title 3000	TV Show	Thriller	2015	PG	2	United States

3000 rows × 7 columns